ISSN 0956-9979 FEBRUARY 1997

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

IN THIS ISSUE:

• **A sheltering sky?** Continuing the occasional series on viruses around the world, this month we head north to Scandinavia. See p.15 for details of the situation there.

• **A Wordy topic.** The past eighteen months have seen the growth of what has become the most often-reported virus type in the world – the macro virus. Turn to p.16 for the *VB* tutorial.

• **Howling for recognition.** The author of the WereWolf family has created a large variety of widely-divergent types of viruses, ranging from direct-action to resident; from stealth to polymorphic. Igor Muttik's analysis begins on p.12.

CONTENTS

EDITORIAL

Office Conversion

“ viruses converted to VBA5 will not be detected by current anti-virus products ”

The timing of this year's *NCSA* International Virus Prevention Conference in Washington DC (16/17 January) was a double-edged sword. On the one hand, it was close to Christmas and the New Year, making it difficult to complete presentations on time; on the other, it coincided with ceremonies for the presidential inauguration, giving me the chance to watch the terrific fireworks display on the evening of the 18th. There was another hidden bonus to these dates: on 16 January, *Microsoft's* latest version of *Office* arrived in North American shops, which allowed the many European anti-virus people present the opportunity to pick up a copy (it is not yet available in Europe). Amazingly, to buy their copies, they had to make it through queues of ordinary citizens at the Washington software stores. A few years ago the idea of normal people lining up to purchase such a piece of software would have been inconceivable: *Windows 95* changed all that, starting a trend continued by *Office 97*. *Microsoft* has enough marketing push to ensure that people will rush out and buy major new products, even at around $500 a throw (for the Professional version).

Office 97 includes new versions of the *Office* applications, all of which now offer the same programming language. *Visual Basic for Applications* version 5 (*VBA5*) is extremely powerful, rich in features and complexity, object-oriented (a buzzword which seems important in marketing these days…), and above all totally different from *WordBasic*.

This fact in itself presented *Microsoft* with a problem – like any software company releasing a new version of its software on which it has spent millions of dollars, it must both provide a compelling reason to upgrade, and maintain backwards compatibility. The importance of the second point should not be underestimated: not providing this would doom the new product to failure, especially if it has as overwhelming a user-base as the *Office* applications.

One aspect of the backwards-compatibility problem interests me particularly – macros. *Word 6.0* and *7.0* macros must be seamlessly converted to *VBA5* macros as old documents are moved to *Word 8.0* (the version of *Word* in *Office 97*). This is a one-time process, but would seem to imply that virus macros would be automatically upgraded to *VBA5*…

Fortunately, the reality is not that bad: *Microsoft* has done two things to lessen the problems. First, certain recognised *WordBasic* macro viruses (at the moment, these seem to consist mainly of Concept and Wazzu variants) will not be converted to *VBA5* when infected documents are loaded. Second, perhaps more important in the long run, when a document containing *WordBasic* macros is loaded, *Word 8.0* by default pops up a message box informing the user that the document may be infected. *Word 7.0a* does this too, and like *7.0a*, *8.0* does not display the warning if the document is loaded from the templates directory. The default option is to load the document but disable the macros.

Steve White, of *IBM's* TJ Watson Research Labs, gave a talk at *IVPC97*: it focused in part on what might happen to the spread of macro viruses as *Office 97* becomes more widely used. He believes their prevalence will decline, due to the ‘anti-virus’ features of *Word 8.0* mentioned above – I believe he's right. Users without anti-virus software will have document infections removed by *Word*.

However (the ubiquitous ‘however’…), viruses converted to *VBA5* will not be detected by current anti-virus products. *VBA5* macros, as befits a new language, are stored in a different manner from *WordBasic* macros, which means anti-virus companies must put in more work to determine how to detect viruses in, and remove them from, the *VBA5* section of files generated by the new *Word*. Coming so soon after the previous tremendous effort to do the same for viruses in the *WordBasic* section of *Word* files, this fact must be at the least depressing, and more likely downright annoying.

This sort of thing has happened before. Anti-virus companies have had to build in an understanding of the formats of NE (as used by *Windows 3.1*) and PE (as used by *Windows 95* and *NT*) files. However, these are not as complex as those used by *Office*, and the need for the understanding was less pressing in those cases. This time, the situation appears more risky, and the file format more complex… once again, vendors must play catch-up for all they're worth – or they won't be worth all that much.

NEWS

Results all Round

A quick-fire burst of financial results peppers late January: anti-virus vendors *Dr Solomon's*, *McAfee*, and *Symantec* have all announced results for late 1996 quarters.

The happiest of the money men must surely be at *McAfee*: this company announced its results for the fiscal year ended 31 December 1996. Fourth quarter revenue alone reached almost US$60 million (more than double that in the same period a year before), raising post-tax profit (net income) for the quarter to in excess of US$15 million (up from US$7.3 million for the same period the year before).

Performance over the fiscal year was similarly startling: revenue doubled from US$90 million to US$181 million, boosting post-tax profits to an impressive US$39 million. Shareholders also have reason to be cheerful – earnings per share rise to US$0.92 over the year.

Symantec exhibited a far lower profit/revenue ratio: they booked income of US$124 million in the same quarter, more than twice that of its biggest competitor in anti-virus software. However, the company's net income for the quarter was just under US$14 million – lower than that of *McAfee*! This reflects the fact that *Symantec* is a far more diversified company than *McAfee*, with its wider range of products that stretches from *Norton AntiVirus* to *pcANYWHERE*, from *WinFax PRO* to *Visual Café*. Earnings per share are set at US$0.25.

The second quarter for *Dr Solomon's Software* ended with November 1996: their statement is slightly more complex, but still shows strong growth. Bookings increase by over 75%, to just under US$17 million, and turnover (this figure allows for some of the bookings to be deferred to subsequent quarters) is up 90% to just over US$14.5 million. Profits jump to around US$3.2 million in the quarter, from under US$0.7 million in the same quarter twelve months ago.

Inevitably, therefore, one is left wondering who did best. In terms of sheer magnitude of figures, *McAfee* takes the honours. In terms of the profit/revenue ratio, they are closely followed by *Dr Solomon's Software*. The *McAfee* figures show that 25% of its income is profit. The *DSS* figures are slightly more opaque, but here it is around 20%; *Symantec* comes in third at 11%.

With the number of corporate PCs increasing rapidly, and the current high level of awareness of the importance of anti-virus protection, the immediate future for anti-virus companies and their shareholders looks rosy.

The share prices for all of the companies discussed above jumped at around the time of the announcements – the interest of the market in anti-virus companies will have increased to match this ∎

Prevalence Table – December 1996

Virus	Type	Incidents	Reports
Concept.A	Macro	54	21.0%
Form.A	Boot	21	8.2%
NPad	Macro	21	8.2%
AntiCMOS.A	Boot	16	6.2%
Parity_Boot.B	Boot	15	5.8%
Ripper	Boot	10	3.9%
Junkie	Multi	9	3.5%
Wazzu.A	Macro	9	3.5%
AntiEXE.A	Boot	8	3.1%
WelcomB	Boot	7	2.7%
Empire_Monkey.B	Boot	6	2.3%
Sampo	Boot	6	2.3%
Stoned.Angelina	Boot	6	2.3%
MDMA	Macro	5	1.9%
NYB	Boot	5	1.9%
Manzon.1414	File	4	1.6%
Die_Hard	File	3	1.2%
Empire.Monkey.A	Boot	3	1.2%
Quandary	Boot	3	1.2%
AntiCMOS.B	Boot	2	0.8%
Hassle	Macro	2	0.8%
Jumper.B	Boot	2	0.8%
One_Half.3544	Multi	2	0.8%
StealthBoot.C	Boot	2	0.8%
TaiPan.438	File	2	0.8%
Telefonica	Multi	2	0.8%
Tentacle.10634	File	2	0.8%
Other[1]		30	11.7%
Total		257	100%

[1] The Prevalence Table includes one report of each of the following viruses: Azusa, Bandung, Boot.437, Bye, Cascade.1701, Colors.B, Concept.D, Da'Boys, Dark_Avenger.1800, Defo, DelCMOS.B, Divina.B, Edwin, Havoc.3072, Imposter, Junkie, Laroux, Major.1644, Natas.4744, Ornate, Rhubarb, Russian_Flag.A, Satria.A, Stoned.NOP, Tequila, TPVO.3783, Tubo, Unashamed, V-Sign, and Wazzu.I.

VB'97: Abstracts

Plans for *VB* conference are being finalised as this issue goes to press – more complete information will be available in time for next month's edition. In the meantime, preliminary abstracts are welcome, and should be submitted to Alexandra Hothersall (ah@virusbtn.com). Further information is also available from *Virus Bulletin* offices; Tel +44 1235 555139, fax +44 1235 531889 ∎

IBM PC VIRUSES (UPDATE)

The following is a list of updates and amendments to the *Virus Bulletin Table of Known IBM PC Viruses* as of 21 January 1997. Each entry consists of the virus name, its aliases (if any) and the virus type. This is followed by a short description (if available) and a 24-byte hexadecimal search pattern to detect the presence of the virus with a disk utility or a dedicated scanner which contains a user-updatable pattern library.

Type Codes

- **C** Infects COM files
- **D** Infects DOS Boot Sector (logical sector 0 on disk)
- **E** Infects EXE files
- **L** Link virus
- **M** Infects Master Boot Sector (Track 0, Head 0, Sector 1)
- **N** Not memory-resident
- **P** Companion virus
- **R** Memory-resident after infection

BadCom.557

CN: An appending, 557-byte direct infector containing the encrypted texts '*.com' and '\command.com'. Infected files start with 90h (instruction NOP).

```
BadCom.557          B440 BA00 0159 03D1 51B9 2D02 CD21 BF01 0359 03F9 51B9 2600
```

BPU.2269

CER: A stealth, partially-encrypted, 2269-byte virus. The time-stamp on infected files is set to 62 seconds.

```
BPU.2269            81EE 0D00 8BFC 8D1E 2200 BC40 0031 2043 434C 75F9 CBE7 C406
```

Champaigne.508

CN: An encrypted, appending, 508-byte virus which infects one file at a time and contains the texts '*.com', '????????COM', 'ChaMpaIgne v2.oo Lives...', and 'by KrACkBaBy'. Infected files are marked with byte 57h ('C') at offset 0003h.

```
Champaigne.508      8DB6 1301 3E8B 96C1 02B9 D700 3114 4646 E2FA C3
```

Champaigne.523

CN: An encrypted, appending, 523-byte virus which infects one file at a time and contains the texts 'gREETZ tA WarBlaDE...', 'ChaMpaIgne v2.oo.cby KrACkBaBy', '*.com' and '????????COM'. Infected files are marked with byte 57h ('W') at offset 0003h.

```
Champaigne.523      8DB6 1301 3E8B 96D0 02B9 DE00 3114 4646 E2FA C3
```

Champaigne.527

CN: An encrypted, appending, 527-byte virus which infects one file at a time and contains the texts '*.com', '????????COM', and 'ChaMpaIgne v2.oo.b by kRaCkBaBy'. Infected files are marked with byte 58h ('X') at offset 0003h.

```
Champaigne.527      8DB6 1301 3E8B 96D4 02B9 E000 3114 4646 E2FA C3
```

Champaigne.542

CN: An encrypted, appending, 542-byte virus which infects one file at a time and contains the texts '*.com', '????????COM', 'ChaMpaIgne v2.oo.a Lives!', and 'wRITTEN bY kRaCkBaBy!'. Infected files are marked with byte 43h ('C') at offset 0003h.

```
Champaigne.542      8DB6 1301 3E8B 96E3 02B9 E800 3114 4646 E2FA C3
```

Champaigne.585

CN: An encrypted, appending, 585-byte virus which infects one file at a time and contains the texts '*.com', '????????COM', '!!!yOU!cANNOT!sEDATE!aLL!tHE!tHINGS!yOU!hATE!!!', 'ChaMpaIgne v2.oo Lives...by kRaCkBaByGreetz go out ta WarBlaDE!!!'. Infected files are marked with byte 43h ('C') at offset 0003h.

```
Champaigne.585      8DB6 1301 3E8B 960E 03B9 FD00 3114 4646 E2FA C3
```

Gingerbread/Rainbow

CERMD: A family of multi-partite viruses containing the text: 'HiAnMiT' and '*4U2NV*' (variant 2073 does not have the latter). Rainbow variants contain the string: 'roy g biv'; Gingerbread variants contain two messages: 'Made in OZ' and 'You can't catch the Gingerbread Man!!'. All templates detect viruses in infected files and memory.

```
Rainbow.2073        B8AD 1BCD 133D EDDE 754A 9090 0E1F 81C6 D506 813C 4D5A 740C
Rainbow.2249        B8AD 1BCD 133D EDDE 7545 0E1F 81C6 9807 813C 4D5A 7409 BF00
Rainbow.2268        B8AD 1BCD 133D EDDE 754A 9090 0E1F 81C6 9807 813C 4D5A 740C
Gingerbread.2314    B8AD 1BCD 133D EDDE 7546 0E1F 81C6 A307 813C 4D5A 7409 BF00
Rainbow.2350        B8AD 1BCD 133D EDDE 7545 0E1F 81C6 FB07 813C 4D5A 7409 BF00
Rainbow.2400        B8AD 1BCD 133D EDDE 7545 0E1F 81C6 9E07 813C 4D5A 7409 BF00
Gingerbread.2449    B8AD 1BCD 133D EDDE 7546 0E1F 81C6 2708 813C 4D5A 7409 BF00
Gingerbread.2467    B8AD 1BCD 133D EDDE 7546 0E1F 81C6 A907 813C 4D5A 7409 BF00
Rainbow.2471        8DB6 7408 8B04 3D4D 5A9C 741A 4075 10BF 0600 03F7 FF34 8BDC
Rainbow.2501        B8AD 1BCD 133D EDDE 7545 0E1F 81C6 0108 813C 4D5A 7409 BF00
Rainbow.2564        8DB6 D108 8B04 3D4D 5A9C 741A 4075 10BF 0600 03F7 FF34 8BDC
Gingerbread.2567    8DB6 7C08 8B04 3D4D 5A9C 7426 4075 1C0E 07BF 0600 03F7 A5AD
Gingerbread.2602    8DB6 7C08 8B04 3D4D 5A9C 7426 4075 1C0E 07BF 0600 03F7 A5AD
Gingerbread.2714    B8AD 1BCD 133D EDDE 7546 0E1F 81C6 2D08 813C 4D5A 7409 BF00
Rainbow.2626        8DB6 7E08 8B04 3D4D 5A9C 741A 4075 10BF 0600 03F7 FF34 8BDC
```

```
Gingerbread.2692    8DB6 F908 8B04 3D4D 5A9C 7426 4075 1C0E 07BF 0600 03F7 A5AD
Rainbow.2715        8DB6 D708 8B04 3D4D 5A9C 741A 4075 10BF 0600 03F7 FF34 8BDC
Gingerbread.2848    8DB6 FF08 8B04 3D4D 5A9C 7426 4075 1C0E 07BF 0600 03F7 A5AD
```

Green.1131

CN: A prepending, 1131-byte fast direct infector containing the texts 'COMMAND.COM', '*.com', 'PATH' and '????????COM'. The virus does not replicate on systems with a BIOS containing the following string at address F000h:8078h: '40-P101-001437-00101111-072594-GREEN' (probably to avoid infecting the author's machine). It encrypts the host file and stores it inside the virus' body (near the end of the code).

```
Green.1131          B440 8B1E 5904 B939 0490 BA00 01CD 212E 8F06 3705 B802 428B
```

Into.685

CER: A prepending, 685-byte virus containing the texts 'COMEXEexecom' and '[IntOv]'.

```
Into.685            8BFA B000 B966 06F2 AE83 EF04 5850 3D00 4B74 3180 FC3D 740E
```

June8th.1919

CER: A stealth, encrypted, appending, 1919-byte virus containing the text 'EXECOM'. A fast infector, it infects all files in the specified directory while executing the 'dir' command. The virus contains a payload which triggers on 8 June and includes a procedure to overwrite 1280 sectors of drive C.

```
June8th.1919        C08E C0FA 8D86 3A01 26A3 0400 268C 0E06 000E 8D86 5501 50CF
```

Katy.4080

CER: An encrypted, stealth, 4080-byte virus containing the text '[KatyDid Tar] Nostradamus / NuKE'. It corrupts EXE files and crashes the system during the infection attempt. The virus contains an ANSI format message with a large graphic 'VLAD' logo and the following text: 'The joke of Interpol £100 reward by M5 for the person who can successfully connect this Brisbane, Australia, based virus writing group to the recent wave of viruses found in Britian's economic community. If you have any information, send it to frisk@isle.com, and it will be dealt with in the manner of utmost confidentiality! Constable Alan Solomon New Scotland Yard, CCU'.

```
Katy.4080           008D BE45 0033 D2B8 1218 CD21 2E32 8632 0086 E0E8 0B00 EB13
```

Palma.247

CN: An overwriting, 247-byte, direct infector. It infects one file at a time and contains the string: '????????.COM'.

```
Palma.247           B440 BA00 01B9 F700 CD21 B43E CD21 2EA0 DA01 3C02 7505 2EFE
```

Palma.463

CN: An appending, 247-byte, direct infector which infects one file at a time and contains the strings '*.com' and 'dflbv!'.

```
Palma.463           B440 8BD5 81C2 0401 B9CF 01CD 212E 8F45 022E 8F05 B43E CD21
```

Palma.503

CN: An appending, 503-byte, direct infector containing the texts 'Palma Ver.2X', '*.com' and 'dflbv?'.

```
Palma.503           B440 8BD5 81C2 0401 B9F7 01CD 218F 4502 8F05 B43E CD21 B40E
```

Palma.591

CN: An appending, 591-byte, direct infector containing the texts 'Palma Ver.4X', 'C:\COMMAND.COM', '*.com' and 'dflb4?'. On 15th, 16th and 17th of a month the virus deletes the file COMMAND.COM.

```
Palma.591           B440 8BD5 81C2 0401 B94F 02CD 213E C686 2F03 0190 8F45 028F
```

Palma.642

CN: An appending, 642-byte, direct infector containing the texts 'Palma Ver.3X', '*.com' and 'dflb3?'.

```
Palma.642           B440 8BD5 81C2 0401 B982 02CD 213E C686 7103 0190 8F45 028F
```

SayNay

CN: Two minor variants of an appending, fast, direct infector. They contain the texts 'saynay.asm', '*.co?', 'saynay.bat' and 'Magic!'. Infected files have the text 'SayNay' at offset 0003h.

```
SayNay.1515         FAE8 4F01 3E8B 6E00 81ED 0D01 FB8D B697 02BF 0001 B909 00F3
SayNay.1516         FAE8 5001 3E8B 6E00 81ED 0D01 FB8D B698 02BF 0001 B909 00F3
```

Sunnyvaley.2288

CR: An appending, 2288-byte virus containing various encrypted messages 'borrowed' from other well-known viruses (e.g. PCOgre, Stoned, Aids). The virus contains a payload which triggers on Sunday 12 January, 11 February, 10 March, etc, and formats disks. Infected files' time-stamps are set to 62 seconds.

```
Sunnyvaley.2288     8CC3 891E AD09 8D16 0001 B821 25CD 2107 06BB 2C00 268B 078E
```

Tourist.1871

CN: An appending, 1871-byte direct infector containing various texts.

```
Tourist.1871        E94D 0253 B801 25BA 3F01 9003 D3CD 215B 53B8 0325 8BD3 81C2
```

XXX.1060

CR: A prepending, 1060-byte virus containing the text: 'My dear xxx: Happy birthday and happy a new year ! Yours xxx .'

```
XXX.1060            B8EE FFCD 213D FFEE 744E 2EC6 06DC 04FF 90B4 2ACD 2180 FA13
```

YZ.1230

CR: An appending, 1230-virus containing the text: 'iT·IS·MY·BIRTHDAY'. The virus code includes a procedure which overwrites the first 96 sectors of the first physical hard disk. All infected files end with the characters 'YZ' (5A59h).

```
YZ.1230             9C80 FCEF 7505 B85A 4D9D CF80 FC35 750B 3C21 75E0 2EC4 1E20
```

YZ.1339

CR: An encrypted, appending, 1339-byte virus containing the texts 'iT·IS·MY·BIRTHDAY' and 'COMMAND.COM'. The virus code includes a procedure which overwrites the first 128 sectors of the first physical hard disk. All infected files end with the characters 'YZ' (5A59h).

```
YZ.1339             14FC 8DB7 6201 B2?? 8D8F 0506 2BCE 2E30 1446 E2FA FBC3 595A
```

YZ.1434

CR: A stealth, encrypted, appending, 1434-virus containing the texts 'iT·IS·MY·BIRTHDAY' and 'COMMAND.COM'. The virus code includes a procedure which overwrites the first 128 sectors of the first physical hard disk. Infected files end with the characters 'YZ' (5A59h), and have their time-stamps set to 62 seconds.

```
YZ.1434             14FC 8DB7 6401 B2?? 8D8F 6406 2BCE 2E30 1446 E2FA FBC3 595A
```

VIRUS ANALYSIS 1

Shelling Out

Peter Szor
Data Fellows Ltd, Finland

Shell.10634 was found in the wild in June 1996 in USA, UK, Australia, Norway and New Zealand. Like another, related, virus [*see 'Touching the Tentacle', VB, September 1996, p.11*], it was distributed via the Internet several times. One such incident occurred on 3 August 1996, when an infected screen saver called PCTRSHOW.ZIP was posted to alt.sex.pictures and alt.binaries.pictures. (Note that there are also clean copies of PCTRSHOW in circulation.)

There are many similarities between the two viruses: in fact, it is probable that the same person wrote both viruses. Shell has a different infection technique than Tentacle, making Shell an interesting specimen. A non-resident direct action infector, it infects only *Windows 3.x* EXE and SCR (screen saver) programs which are of New Executable (NE) format.

There are as yet very few *Windows* viruses, but each example thus far has changed the Entry Point field in the NE header to take control: Tentacle did this. With Shell, the Entry Point address does not point to the start of the virus code, but to the original Entry Point of the host program. The virus' most interesting feature is that, rather than changing this address, it patches the host program's Segment Relocation Records to pass control to the virus code.

Infection

When an infected program is run, the virus takes control. It then searches for *Windows* EXE programs in the current directory. The virus infects any uninfected NE file, then goes to the directories C:\WIN, C:\WINDOWS, C:\WIN31, C:\WIN311, and C:\WIN95, attempting to infect one file in each directory except C:\WINDOWS. Here, the virus infects two *Windows* programs with EXE extensions. Next, it infects one file with a .SCR extension in the current directory. All these directory names are encrypted in the virus body, thus are not visible by viewing the infected file.

Before the virus infects a file, it checks for the read-only attribute, which it clears if set. Then it opens the victim and reads the EXE header: it does not infect the file if the MaxMem field (word at offset 0Ch in the EXE header) is not FFFFh.

Next, it searches for the host program's NE header. If found, the virus creates C:\TENTACLE.$$$, a hidden temporary file – in Tentacle, this text was visible. Shell decrypts this string, then starts to copy the victim's code into the temporary file. While copying, the virus modifies the NE header fields, creates a new Segment Table which describes a new segment (this will hold the virus code), and writes itself to the end of the file. When the infected copy of the original file is ready in C:\TENTACLE.$$$, the virus copies this file over the original. Finally, it returns the host program's date- and time-stamp back to that of the original, and deletes the temporary file. Infected files will increase in size by 10634 bytes.

Taking Control

Given that the virus does not change the original Entry Segment IP fields of the NE header, one might well wonder how it receives control on execution. Shell searches the Module Reference Table (MRT) for the strings KERNEL and VBRUN300. If neither is found, the virus terminates infection and deletes the file C:\TENTACLE.$$$.

Figure 1: This is the error box which is usually seen when an NE virus infects WINHELP.

Otherwise, the virus picks up the Module Number of the found module name and reads the Segment Relocation Records of each Segment. If it found KERNEL in the MRT, it looks for Relocation Record 91 (INITTASK); if VBRUN300 was found in the MRT, it searches for 100 (THUNKMAIN). Both Relocation Records point to standard initialisation code which must called at the beginning of a *Windows* application.

For example, if KEYVIEW.EXE (a small *Windows* program) is not infected, it has a Relocation Entry for KERNEL.91 for its first segment. When this program is infected, the virus patches this record to point to the virus segment VIRUS_SEGMENT. Thus, the infected file starts as it did before the infection, but when the application calls one of the above initialisation codes, control passes to the virus.

Within the VIRUS_SEGMENT are three Relocation Records, one of which points to the original initialisation procedure KERNEL.91 or VBRUN300. Thus, the virus can start the host program after itself. This is new in *Windows* infectors, and means that Shell is an anti-heuristic *Windows* virus.

Targeting WINHELP

When the virus infects WINHELP.EXE, it patches one byte in WINHELP's second segment. A conditional jump (74h) instruction is replaced by a jump short (EBh) instruction. This technique is typical in cracking; I had to investigate for some time and partially reverse-engineer WINHELP.EXE to understand the idea. Finally I deduced that WINHELP has a procedure to calculate a checksum for itself. This is only a partial checksum, as it does not include every byte of the program. This procedure returns with zero when it does not detect changes in the code, and one if it does.

Shell replaces the conditional jump; thus the message box (Figure 1) will not be displayed, and the user will not notice the infection. As described in the article 'Touching the Tentacle' [*see previous reference*]: 'The most noticeable victim was WINHELP, one of the most-often-used *Windows* applications, which stopped working completely.' The fact is that the virus infected this file correctly, but WINHELP detected the change, displayed a message box for the user, and terminated, just like most programs which have a self-check routine.

It is possible that the author did not previously recognise this problem because only some WINHELP versions have this feature. *Microsoft* added the self-check function only in *Windows 3.1*.

Both Tentacle and Shell are direct action, non-memory-resident viruses, thus the virus writer could be happy to see that his virus spreads whenever a user hits F1. For this reason he did not simply avoid infecting WINHELP, but patched it. Not only WINHELP has the self-check feature – other programs, such as *Microsoft Mail* and *Microsoft Schedule+*, have it as well; thus the user will notice an infection. The problem is that these programs are not as widely used as WINHELP.

Trigger Routine

When Shell completes infection, it checks the time, and triggers if it is between 01:00 and 01:05. The virus creates the file C:\TENTACLE.GIF with the hidden, system, and read-only attributes set, and starts to manipulate the *Windows* Registry. (TENTACLE.GIF is 7875 bytes long, which accounts for the large size of the virus.) To do this, the virus modifies the Imported Name Table by adding a SHELL entry (SHELL.DLL) during infection.

The VIRUS_SEGMENT's Segment Relocation records refer to two standard functions, REGQUERYVALUE and REGSETVALUE. First, the trigger calls REGQUERYVALUE to check the \SHELL\OPEN\COMMAND line under the .GIF extension section. That refers to the command executed to view a .GIF file.

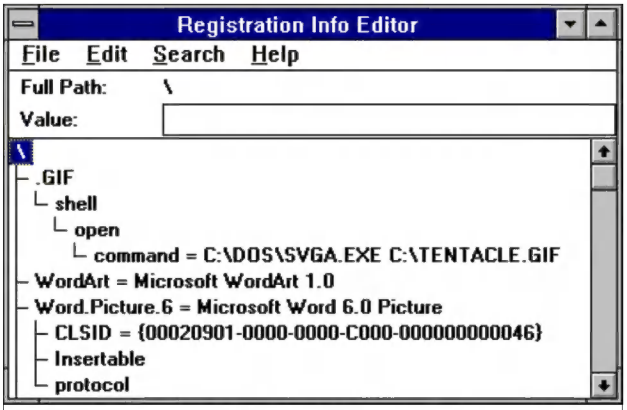

Figure 2: Modification to the registry means that TENTACLE.GIF will be used if a user double-clicks on any GIF file.

Figure 3: This picture will be shown after infection when any GIF file is viewed.

Then the virus replaces the %1 parameter at the end of this line with the line C:\TENTACLE.GIF by using the function REGSETVALUE. For example, the string 'C:\DOS\SVGA %1' will be replaced by 'C:\DOS\SVGA C:\TENTACLE.GIF' (Figure 2).

As a result, the system will show C:\TENTACLE.GIF when any GIF file is clicked (Figure 3). To the user, it looks as though the virus has overwritten all GIF files with its own picture. The virus' second trigger returns the registry to its original configuration from 01:15 till 02:00.

Conclusion

Like Tentacle, Shell has several bugs, which makes it easier to find. However, this virus is one of the most successful *Windows* viruses, and without carrying out a complete disassembly of this virus, it is simply impossible to disinfect it correctly.

Shell.10634

Aliases:	Tentacle_II, Tentacle.10634
Type:	Non-resident, direct action infector.
Infection:	Windows 3.x applications in New Executable format.
Self-recognition in Files:	FFFEh in MaxMem field in MZ EXE header.
Hex Pattern:	`7BCD 210F 823E 01B4 40B9 6429 BA00 001E 0E1F CD21`
Trigger:	Time between 01:00 and 01:05.
Payload:	System displays C:\TENTACLE.GIF image while viewing any GIF file.
Removal:	Delete infected file; replace with known clean copy.

VIRUS ANALYSIS 2

Batalia6.BAT: Polymorphism Conquers the World

Eugene Kaspersky

The first really polymorphic virus appeared in spring 1992; an MtE-based, COM infector named Pogue. In the five years since then, polymorphism has evolved into one of the most popular 'features' in virus-writing, spreading not only to other types of executable files, but also to other platforms. Viruses exist which are polymorphic in EXE files, in SYS files, in boot sectors, and even in memory.

Some time ago, a new type of virus appeared, using a polymorphic engine when infecting the NE (New Executable) files used by *Windows 3.x*. Most recently of all, the first mildly-polymorphic *Word* macro viruses appeared. It can only be a matter of time before we see the first polymorphic *Windows 95*, *OS/2*, and *Excel* spreadsheet viruses.

However, it is not yet time to forget good old DOS – after all, the virus writers haven't. There was one more type of executable that had never had a polymorphic virus to call its own – the DOS batch (BAT) file. This was up until the appearance of Batalia6: with that, polymorphism finally saturated DOS.

Inside the Code

Batalia6 appears harmless – it contains no destructive code, and its only side effects are DOS error messages. Such messages appear if ARJ.EXE (one of the most popular archivers) is not present in the DOS PATH. Since the virus uses only DOS command-line instructions, it is not memory-resident; indeed, it cannot access system memory directly in any way.

When an infected batch file is executed, the virus first searches for other batch files in the current directory, and infects one. During infection, the virus uses the ARJ archiver first to extract its data files, and then to re-pack them into the newly-infected file. If ARJ.EXE is not found in a directory in the DOS PATH, Batalia6 fails to replicate.

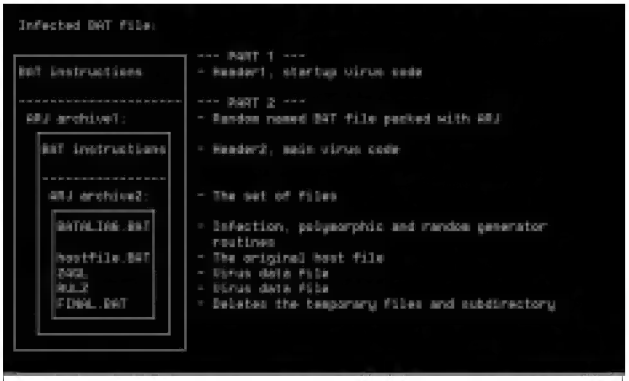

Figure 1: An infected file resembles a BAT file merged with an ARJ archive, within an ARJ archive, merged with a BAT file. [*Phew! Ed.*]

The virus' main feature is its polymorphism in BAT files: seeing a polymorphic BAT virus that functions in this way is unexpected. Several extant BAT viruses combine DOS commands (the BAT part) and binary executable data (the executable part). These viruses all work in the same way: by renaming the host BAT file to COM, and running it as a binary executable. It would perhaps have been less surprising had the first polymorphic BAT virus been one of these. Batalia6, however, is a different type of beast.

Infected Files

Like the BAT viruses described above, files infected with Batalia6 contain two parts – BAT commands and data. Unlike those viruses, however, the data does not take the form of assembled code; rather, it is an archive.

The first part (the header) contains five DOS commands; the second part (the remainder) is a password-protected ARJ archive, containing a randomly-named BAT file. As a result, infected files contain both text strings (DOS commands) and binary data (ARJ archive).

The randomly-named BAT file contained with the outermost ARJ archive is also in two parts – again, virus code in the form of batch commands, and compressed data. The compressed data contains the host file, two more virus code files, and two data files. For a diagrammatic representation of the overall file, see Figure 1.

The Infected BAT File

Header1 consists of five lines (three of these are commands, and two of them are junk) selected from a range of options with varying lengths. This, coupled with the fact that the ARJ archive is encrypted with a randomly-selected password, means that the virus does not contain any constant bytes.

When executed, the virus (commands in Header1) disables DOS echoing (first command), runs the ARJ archiver to extract the BAT file from the second part of the host file (third command), and executes the newly-extracted BAT file (fifth command). The second and fourth commands are junk code which do not influence the program's flow in any way.

To extract files from the ARJ archive, the virus uses the fact that ARJ.EXE accepts input files where the compressed data is not located at the beginning, but has other data (in this case, the batch commands) prepended to it. It is not known how far the archiver is willing to look for relevant data, but it is far enough for Batalia6.

Figure 2: These comments, among others, can be found lurking within the code of Batalia6.

Preparing the File

When extracted and executed, the code in Header2 creates a temporary directory; extracts the files from the inner archive into the new directory; executes the host file; calls the searching, infection, and polymorphic routines; and finally deletes the temporary files and directory. This section of the virus is not polymorphic – it always consists of the same 21 BAT commands.

The files Header2 unpacks from the inner archive are:

- BATALIA6.BAT: infection and polymorphic routines
- FINAL.BAT: erases temporary files and directory
- RULZ: BAT file containing the commands in Header2 (used whilst infecting a new host)
- ZAGL: random data used by the polymorphic engine
- Hostfile.BAT: the original host file

Search and Infect

When BATALIA6.BAT is run, it searches for *.BAT files in the current directory using the following DOS command:

```
for %%b in (*.bat) do call s_g_w_w\%0 f %%b
```

This command locates all available batch files, and for each one, re-runs BATALIA6.BAT with two arguments: 'f', and the name of the batch file (s_g_w_w is the name of the temporary directory into which the files were unpacked).

When the file BATALIA6.BAT is executed with 'f' as the first argument, it will check the file given in the second argument to see whether or not it is already infected – it does this by executing ARJ with the command option 'l'. This option on ARJ instructs it to list the contents of an archive. The virus then checks the value of ERRORLEVEL, which has been set by ARJ depending on whether or not it found a valid archive.

If the file is already infected, ARJ.EXE will find an archive within the batch file, and will set ERRORLEVEL appropriately. If it is not already infected, there will be no archive within the file, and ARJ will set ERRORLEVEL to a different value. BATALIA6.BAT uses this to branch its execution appropriately.

If the file is not infected, Batalia6 runs its polymorphic engine. This creates an image of an infected file and packs it using ARJ. When preparing and compressing files, the virus uses a variety of fairly complex constructions.

A random number generator is used to select which of the available lines should be used in this particular instance of the virus. Depending on the values returned by the generator, the virus selects and stores DOS commands to Header1, with each command selected from four variants.

The virus has a clever way of running its random generator: it creates three files, called RND1, RND2 and RND3, and writes data to each of them. Then it creates a file called TEST, and executes the following DOS command:

```
del rnd? /p <test>nul
```

The DEL command has prompting turned on (if this were executed from the command line, the user would have to confirm the deletion of each of the three files in turn), and uses the data in TEST as input. Finally, output is redirected to NUL.

The result of all this is that which RND? files are deleted depends entirely on the symbols contained within the TEST file. The random generation routine is then executed according to all remaining RND? files (that is, those which were not deleted).

Last Notes

I had never realized the DOS batch language was so powerful. Many algorithms, tricks and even a polymorphic engine may be (and indeed have been!) programmed, using a relatively small set of DOS commands. It is only a pity that this functionality (and the ability of the author) has been directed merely to writing new viruses.

Batalia6

Aliases:	None known.
Type:	Non-memory-resident, polymorphic BAT virus. Parasitic, but uses new technology.
Infection:	Batch files only.
Self-recognition in Files:	Runs ARJ.EXE to check archive within a file and checks ERRORLEVEL; see analysis for details.
Hex Patterns in Files:	Virus is polymorphic; see analysis for text strings.
Trigger:	No trigger routine.
Prevention:	Remove ARJ.EXE from PATH.
Removal:	Remove and replace infected files.

VIRUS ANALYSIS 3

Alien: Definitely a Native!

Neville Bulsara
N&N Systems and Software, India

At the end of November, we had a frantic call from a client who was encountering all sorts of unexpected messages while running *MS Word*. He was sure it was a virus, but his anti-virus software was detecting nothing.

He sent us copies of some document files and NORMAL.DOT: cursory examination of these revealed the presence of three macros in the document – FileSaveAs, AutoOpen, and AutoClose. All were execute-only and could not be viewed or edited through the ToolsMacro function from within *Word*.

Luckily, barely a month back we had embarked on a project to decipher the structure of *MS Word* documents and had already had some success in that direction. Thus, we were able to point our tools at the documents and decrypt the macros.

Till then, I had suspected that this was a new virus of 'alien' (that is, foreign) origin, and that it had found its way to India via the customary routes. However, we soon discovered that the virus was of Indian origin – this brought back memories of the first breed of Indian viruses: PrintScreen and Joshi. India, it seems, has finally achieved the dubious 'distinction' of producing indigenous macro viruses! Somehow, I do not feel proud.

Infection

Infection of the Global Template is achieved via the virus' AutoOpen macro, while infection of other documents is achieved via both the AutoOpen and the AutoClose macros.

The AutoOpen Macro first calls DisableInput(), to prevent the use of the Escape key to halt macro execution. It then uses the Now() function to check if more than 35,399 days have elapsed since 30 December 1899: this essentially waits until after 1 October 1996. If this condition is satisfied, the virus sets a flag called OkToRun to true.

The virus then checks to see if the current document (or Template) filename contains the string 'ALIEN'. If so, the variable 'Queen' is set to true.

The virus then goes through a complicated procedure in which several flags are set depending on the state of its macros. If the Global Template contains the macros AutoOpen, AutoClose and FileSaveAs, the flags GAO (GlobalAutoOpen), GAC (GlobalAutoClose) and GFSA (GlobalFileSaveAs) are set to true.

Each macro is then checked to ensure it is execute-only – the flags GAOx, GACx, and GFSAx are set if the corresponding macro is execute-only or if the 'Queen' flag is set.

The virus then checks the current document for the presence of the three macros. Variables called LAO, LAC, LFSA are set to true as are their corresponding 'x' (execute-only) flags based on the tests similar to those explained above for the Global Template.

Next, Alien sets a variable called 'GInstalled' (GlobalInstalled) to true if GAO, GAC, GFSA and their corresponding 'x' flags are all true. The variable 'Linstalled' is set based on similar tests on the 'L' flags.

The virus then infects the Global Template if it is not already infected (If Linstalled And Not Ginstalled). This involves deleting extant AutoOpen, AutoClose and FileSaveAs macros from the Global Template. The virus turns off the prompt to save the Global Template (accessed via the Tools/Options/Save menu). Thus, when NORMAL.DOT is saved, the user does not receive a prompt.

For good measure (literally – the virus contains a remark to that effect!), the virus removes the ToolsCustomize and ToolsMacro menu options from the default setup (i.e. that held within the Global Template). This is an attempt at stealth to ensure that the user cannot examine various aspects of the Global Template settings after infection.

If the AutoOpen macro receives control from the Global Template (the virus has infected the global template and the user is opening a document), Alien proceeds to infect. This portion of code is identical to the virus' AutoClose macro.

Alien Humour – the AutoOpen Macro

The virus then proceeds to launch its payload depending on several conditions. The payload is contained in the virus' AutoOpen macro and receives control after the test that checks if the local template (the open document) is infected.

If the date is 1 August, and the OkToRun flag is true (i.e. it is 1 August of any year after 1996), there is a 50% chance of the virus putting up a message box celebrating 'Another Year of Survival'. After this, the virus hides Program Manager (thus, under *Windows 3.x*, *Windows* cannot be shut down) and exits the current document without saving it.

If it is a Sunday after 1 October 1996, there is a 50% chance of the virus displaying a message that it intends to take a sabbatical that day. There is a further 50% chance of the virus hiding Program Manager and yet another 50% chance of it closing the current document without saving it.

There are also small chances that the virus may display one of a range of messages, and a 5% chance that, if the name of the current document is less than nine characters, an advisory is shown: 'Longer File Names Should Be Used.', with the title 'Tip From The Alien'.

The virus code at this point contains a remark 'Behold the Alien Virus!! Lets See how it survives!' and another one stating that it was written in Chandigarh (a city not far from the Indian capital Delhi) on 1 August 1996.

AutoClose

The AutoClose Macro starts off almost identically to AutoOpen (DisableInput, set status of Global and Local flags etc). The rest of the code is identical to that part of the AutoOpen Macro which handles the infection of documents.

The virus checks if the current document (local template) is infected (Linstalled is set). If not, the virus checks for the presence of AutoOpen and FileSaveAs macros in the document under attack. If found, these are deleted, and a variable called SaveTheFile is set to true.

The virus then checks whether or not it is resident in the Global Template. If so, it deletes any AutoClose Macro it finds in the local template. Next, it copies its AutoOpen, AutoClose, and FileSaveAs macros to the local template (the document), and knocks off the ToolsMacro and ToolsCustomize menu options.

The virus then sets the FileSaveAs.Format value to 1, indicating that the file should be saved as a template. The FileSaveAs macro contains only the normal code generated by *Word* when a macro called FileSaveAs is created [*see p.16*].

Summary

The actions of the AutoOpen and AutoClose macros enable the virus to spread rapidly. Alien has been seen in the field at least once. It removes the 'Macro' and the 'Customize' options from the Tools menu; however, these can be restored via the View/Toolbars menu.

The virus is expected to have infected several systems already, due to the two month period before it starts announcing its presence. However, it would seem that it will ultimately limit its spread by revealing its presence in such a visible fashion.

Alien

Aliases: None known.

Type: MS Word macro infector.

Hex Pattern in Files:

```
82E4 86CF DBDA C1C1 DECB C090
EAE1 8DE7 8DC2 CFC1 E08C 8286
E28F 8EEA E18D E789 FDC3 EFED
```

Trigger: Various date-dependant triggers; see text for further details.

Removal: See generic article in this month's issue; turn to p.16.

VIRUS ANALYSIS 4

The WereWolf Family: When Wolves are Cubs

Igor Muttik
Dr Solomon's Software Ltd

In early 1996, the many thousands of extant viruses were joined by a new family. The relationship of its members was immediately obvious – many carried the same identifying string, 'WereWolf', which became the new family's name.

Although the growth in the number of known viruses is about 150-200 per month, it is surprising to see a whole family appearing in such a short period of time. About a dozen new, very different viruses appeared in just three months, presumably from one source. The family also included an unprecedented range of variants; from direct action viruses to fast infectors, stealth and polymorphic. By the end of 1996 there were thirteen WereWolf variants.

Fortunately, no new ones have appeared recently; perhaps now is a good time to take a closer look at this family. The comparative analysis of viruses within a large group often helps trace the tendencies in virus writing, and analyse what inspired the author. WereWolf.1500.b was reported in the wild in many countries, increasing interest in the whole family.

Classification: 'Cubs' and 'Wolves'

Viruses in this family fall into two distinct groups. The first consists of non-resident EXE infectors (658, 678, 684.a, 684.b, and 685). The second includes resident stealth COM/EXE viruses – the latest variants are polymorphic (1152, 1168, 1208, 1361.a, 1361.b, 1367, 1500.a, and 1500.b).

For simplicity, we shall call them 'cubs' and 'wolves', respectively. The list of WereWolves is presented in chronological order in Figure 1 (see right). Let's take a closer look at both subfamilies.

Cubs

The simplest of the WereWolf viruses are direct-action non-resident parasitic viruses (658, 678, 684.a and .b, 685) and infect only EXE files. Their life-cycle is primitive: when an infected file is run, it looks for and infects all suitable EXE files in the current subdirectory. 'Cubs' append themselves to the end of the host file in an encrypted form.

WereWolves use the byte at offset 11h to distinguish between clean and infected files. This is the high byte of the EXE header's SP field, and is set to 06 on infection (i.e. all infected EXE files have SP=600..6FF; normal files with this SP would not be infected). WereWolf's cubs do not infect EXE files beginning 5Ah 4Dh (only with the standard 4Dh 5Ah).

After infecting all possible victims, the viruses find and delete all files named *.MS, *.CPS and ANT*.DAT in the current subdirectory. These are names of files holding integrity-checking information for various anti-virus products.

The cubs are encrypted, but not polymorphic. Rather than polymorphism, the author of WereWolf and its variants clearly set his goal at defeating heuristics. His attempt was quite successful: none of the cubs is picked up either by *TbScan* or by *FindVirus* in heuristic mode.

The trick used to avoid detection is primitive but effective. The virus' very first instruction changes just one byte in its decryption loop, turning the register assignment instruction (MOV AX, const) into a memory modification instruction (XOR [DI], const). Until this is done, the code does not look like a decryption loop at all.

This fools the heuristics, which can see no instructions to change memory, and decides not to run the long do-nothing loop – very effective in the case of EXE files. Moreover, even if the sample WereWolf cub is encrypted with a zero key, heuristic analysis is still a problem: all calls to DOS are done in a manner which is not immediately obvious.

Instead of the normal sequence MOV AX,const/INT 21, the virus first assigns a constant to the BP register. Then, whenever it wants to call DOS, it uses MOV AX,const/XOR AX,BP/INT 21. The heuristic analyser is lost at this point: it

First seen	Variant	Aliases (in italic) and strings
Jan 1996	658	HomeSweat-668 Home Sweap Home (C)1994-95 WereWolf *.s_e *. .. \ *.MS *.CPS ANT*.DAT
	684a	CFangs, Claws-684 *. .. \ *.MS *.CPS ANT*.DAT
	685	CFangs-685, WEREWOLF.693 *. .. \ *.MS *.CPS ANT*.DAT
Feb 1996	678	Werewolf-SweapHome, HomeSweat Home Sweap Home (C)1994-95 WereWolf *.s_e *. .. \ *.MS *.CPS ANT*.DAT
	1208	WereWolf_II, WereWolf.Beast, Were BEAST (C)1995 WereWolf
	1152	WereWolf_III, WereWolf.Scream, WeWo-1152 SCREAM (C)1996 WereWolf
	1367	WereWolf.FullMoon, WeWo FULL MOON (C)1995-96 WereWolf
Mar 1996	684b	CFangs, Claws-684
	1361a-b	WereWolf-FullMoon, WeWo-1152
	1500a-b	WereWolf.Wulf a: WULF, 1996 WereWolf b: [WULF](c) 1995-1996 WereWolf
Jul 1996	1168	WereWolf_III.1168, WereWolf-Scream-1168 SCREAM!(C)1995-96 WereWolf

Figure 1: The numerous variants of the WereWolf family.

just cannot keep track of all the registers (in this case, BP) and does not understand which calls the analysed program is making to DOS.

WereWolf cubs substitute Int 01h to prevent analysis under a debugger. They contain code which was probably meant to be a payload, but does nothing (most likely because of a bug).

Mature 'Wolves'

The mature viruses in the WereWolf family have the following infective lengths: 1152, 1168, 1208, 1361 (a and b), 1367, and 1500 (a and b). They affect COM and EXE files, are all memory-resident, and all have stealth capabilities.

The 1208-byte variant remains somewhat separate from the main group. It is the only fast infector, replicating to files as they are opened. Also, it infects COM files by prepending itself (thus it does not infect COM files shorter than itself). It is not encrypted (as are 1152 and 1168), and marks infected files in a different way.

It seems the author of these viruses did not like fast-infection technology: his later viruses infect files only on execution. The most complex variants (1361.a, 1361.b, 1367, 1500.a, and 1500.b) are slightly polymorphic. The decryption routine is simple, using the stack pointer to decrypt the virus body (preventing direct use of debuggers to decrypt manually). The sequence of operations is always the same – POP reg/XOR reg,word/PUSH reg – but the utilised register varies, making the decryptor variable. This poses no problem for detection: this level of polymorphism is often called oligomorphic, from the Greek 'oligos', meaning small, few.

Installation in Memory

When an infected file is run, it first makes an 'Are you there?' call – this is used by resident viruses to ensure that they install themselves in memory only once.

Variants 1152 and 1168 use AX=0257h, returning DI=100h and CY (this DOS function AH=2 prints a character; the virus calls it with DL=0Dh to type carriage return). 1361 (a and b) and 1367 use AX=3077h, returning DI=100h and a carry flag (CY). WereWolf.1208 uses AX=0287h and returns AX=100h and CY (also emits CR). The most complex variants, 1500.a and .b, use AX=33B3h and return just CY. When the virus is resident, any of these calls return the carry flag (CY). In this case, control transfers immediately to the host file.

If the virus is not resident, the infected file disables the resident anti-virus programs Vsafe and Vwatch (via the usual Int 16h; AX=FA02h, DX=5945h). Then it scans DOS memory for the presence of the memory-resident drivers TbFile, TbMem, TbCheck, and TbDisk, (shareware behaviour blockers from the Dutch company *ESaSS*). If any is found, it is disabled by changing the appropriate part in the memory image of the program. This is why all wolves carry this string of driver names in their body (32 letters; 4 strings of 8 letters each):

```
TBMEMXXXTBCHKXXXTBDSKXXXTBFILXXX
```

After disabling the above-mentioned behaviour blockers, the virus scans the DOS MCB chain, creates its own memory block at the very top of DOS memory, copies its own code there and intercepts vector 21h. WereWolf.1500.a and 1500.b also intercept Int 13h, which is used only for the payload.

Wolves utilise the following Int 21h DOS functions: 11/12h (FindFirstFCB/FindNextFCB), 31h (Terminate and Stay Resident), 3Dh (Open: WereWolf.1208 only), 49h (Wait: not in WereWolf.1500.a or .b), 4Bh (Execute), 4Ch (Terminate), and 4E/4Fh (FindFirst/FindNext).

Now virtually any executed program will catch the virus if it is not yet marked as infected. During infection, wolves hook Int 24h (the critical error handler) to suppress error messages on write-protected media.

Stealth Properties and Infection Markers

The viruses control the execution of programs, paying special attention to CHKDSK.EXE and avoiding any of the following anti-virus programs: *Clean*, *AVP*, *TbScan*, *Scan*, *NAV*, *FindVirus*, *Guard*, and *F-Prot*. WereWolf.1208 also avoids *Quick Basic* (QB). To recognise these programs, the viruses carry the following strings in their bodies (WereWolf.1208 has 'F-PR' instead of just 'F-' and also 'QB'; WereWolf.1500b has 'CHKDSK' rather than 'CHKDS)':

```
CLEAN,AVP,TB,V,SCAN,NAV,IBM,FINDV,GUARD,FV,CHKDS,F-
```

A common problem for all stealth viruses is how to suppress error messages from CHKDSK. When run on a system infected with a stealth virus, CHKDSK reports allocation errors, because the reported sizes of the files do not match their real sizes. This happens because reported file size in bytes does not match the number of clusters allocated in a file allocation table.

All wolves recognise that CHKDSK.EXE is being run and turn off their stealth routine while the check is performed (this is the only reason of controlling DOS functions 31, 49 and 4C; to catch the execution of CHKDSK). They all turn off stealthing if (and only if) you run CHKDSK. For example, if you launch SCANDISK (provided with DOS 6 and above) or rename CHKDSK.EXE to, say, DSKCHK.EXE, you will see file allocation errors caused by a stealth virus.

Many stealth viruses use a file's time-stamp as an infection marker. So do the WereWolves; the seconds field. 1500.a and 1500.b set the seconds field to 6 after infection; all others, to 46. WereWolf.1208 is a special case: this virus divides the date of file creation (usually returned in DX register) by 30 and puts the remainder of the division in the seconds field.

As a result of the fact that all wolves use a legal seconds value as an infection marker, the size of some non-infected files is reported incorrectly. This is because the virus' stealth routine blindly subtracts its own size from that of all files with the 'unlucky' settings. The WereWolf viruses' stealth capability is limited to concealing the size of the infected file. A decent integrity checker would report the modification.

If you inspect files visually, these hints might help isolate samples infected with WereWolf.1500.a and .b: infected EXE files always have IP=10h (word at offset +14h in the EXE header), and all infected COM files finish with eleven zeros followed by four FFh bytes.

Payloads

The wolves' payload is almost the same in variants 1168, 1208, 1361.a, 1361.b and 1367. The payload becomes active immediately after installation into memory.

After infection of every file the virus loads the word from [0:46C] (the system timer) and isolates the six lowest bits (5 bits in 1361.a, 1361.b, and 1367). If they are zero – the probability of this is 1/64 or 1/32 – the virus obtains the date from CMOS clock.

On even dates, these WereWolves write a sector of garbage to the first hard disk in the system; on odd dates, to the second HD. The sector and cylinder where the garbage will be written is coded by year (e.g. in 1997 it would be CX=1997), and the head of the hard disk equals the current month (e.g. in February, DH=02).

Then the virus performs a disk write via Int 13h, which, depending on the hard disk geometry, might fail. On huge hard disks with plenty of cylinders, heads, and sectors per track, the virus may damage data. For example, in 1996/1997 the virus writes to track 537, sector numbers 22/23: these are not present on old, small hard disks. Unfortunately, these sectors will be present on modern HDs, with their 32-63 sectors per track.

WereWolf.1500.a and .b have a more complex payload, which works immediately after infection, and permanently, with no trigger date. These variants intercept Int 13h (disk I/O), which is used to cause damage. The virus' interrupt handler controls disk writes, and when a sector is written to disk, the virus changes a single byte in the buffer, with an approximate probability of 3/100,000.

This number appears low, but the number of errors introduced depends on the amount of written data. If, for example, you are installing new software and write many megabytes of data to disk, the virus will likely cause at least a couple of errors in copied files. This type of damage is usually regarded as the worst, as it is slow and can go unnoticed for a long time. Even backups may be corrupted.

Summary

None of these viruses is remarkable by itself, but the family as a whole was certainly worth a look. I believe they were all written by one person, in a very distinctive style – errors in the source even travelled from one virus to another. First, he created simple direct-action viruses, clever enough to fool heuristics. Then he experimented with resident viruses, with fast infecting and prepending techniques, with stealth and polymorphism.

The WereWolf viruses are well-written; they replicate very reliably in different environments and do not hang. The most recent WereWolves are probably the shortest parasitic stealth polymorphic viruses I have ever seen. I consider the author of WereWolves to have above average virus-writing ability. His skills are extensive – the broad spectrum of created viruses is a clear manifestation of this.

Despite his experience, the WereWolves' author is malicious: each virus has a destructive, immediately active payload. Such persistence is quite rare, especially for skilful virus writers.

WereWolf 658 – 685

Aliases: See Figure 1.
Type: Direct-action.
Infection: EXE files (starting with 4Dh 5Ah only).
Self-recognition:
Byte 06 at offset +11 in an EXE file.
Hex Pattern in Files (near eof):

```
B835 ???? 4747 81FF ??02 72??
C3??
```

Payload: Does not work due to a bug.
Removal: Delete infected files. Replace with known clean copy.

WereWolf 1152 – 1500

Aliases: See Figure 1.
Type: Resident and semi-stealth. Variants 1361.a, 1361.b, 1367, 1500.a, and 1500.b are polymorphic.
Infection: COM and EXE files.
Self-recognition in Memory:
See description.
Self-recognition in Files:
Using seconds field in file time stamp (see details in text).
Hex Pattern in Files (1152, and 1168, and 1208 only; others are encrypted):

```
00BE 1A00 2E80 3E?? ??12 77??
47BE 1D00
```

Hex Pattern in Memory:

```
00BE 1A00 2E80 3E?? ??12 77??
47BE 1D00
```

Intercepts: Int 21h – DOS handler. Int 13h – disk I/O handler (1500.a and 1500.b only). Int 24h (temporary, during infection).
Trigger: None – payload immediately active.
Payload: Writes to the disk at random locations.
Removal: Delete infected files. Replace with known clean copy.

FEATURE

Beneath Northern Skies

Mikko Hyppönen
Data Fellows Ltd, Finland

The Nordic countries (Sweden, Norway, Denmark, Iceland and Finland) have had their fair share of viruses over the years. In fact, one of the very first file-infecting viruses was written in Finland. This was Crew.2480, first seen in 1987. Quite a few viruses have been written in Nordic countries – the long, cold, and dark winters seem to have kept people inside, playing with their computers. [*Ahem. Ed.*]

Sweden has had the biggest problems: several active virus groups have operated from there, including such groups as BetaBoys or Immortal Riot. The result of all this activity is that more viruses have originated in Sweden than in all the other Nordic countries combined.

It is estimated that over two hundred viruses have been written in Sweden, compared to the few dozen from each of the neighbouring countries, and the handful from Iceland. Most viruses written in Nordic countries are DOS-based COM/EXE infectors, with the occasional addition of more advanced viruses, such as Ekoterror, Desperado or Finnpoly.

There are remarkably thorough statistics about the history of the virus situation in Finland, beginning with the first-ever viruses found locally. The statistics show that around 350 viruses have been reported as 'discovered in Finland': some of these were seen only once, and some are found daily. The list of the ten most common viruses is pretty typical:

- WordMacro/Concept
- Form.A
- Finnish_Sprayer
- Parity_Boot.B
- Tai-Pan
- AntiExe
- Finnpoly
- Junkie
- WordMacro/MDMA
- Burglar.1150

Four out of this top-ten list are of Finnish or Swedish origin. If we look at the number of virus reports in Finland during recent years, the picture looks like this:

Year	Reports	New in Finland
1994	456	58
1995	373	30
1996	415	42

Interestingly, the number of virus incidents would have been much smaller in 1996, had macro viruses not appeared. Macro viruses do cause problems in Nordic countries, just like everywhere else. However, more than half of the *Word* or *Excel* installations in use are localized versions, rather than the more vulnerable English versions.

Many *Word* macro viruses are limited to working with the same language version of the application the virus was first created with: for example, Concept fails to replicate under the Finnish or Swedish versions of *Word*. This has a big effect on the spread of these viruses. However, of the 200+ extant macro viruses, dozens do work successfully under any language version of *Word*.

Surprisingly, no macro viruses specific to, for instance, Danish or Swedish versions of *Word* have been found. However, this does not mean no macro viruses have been written in these countries – such viruses could have been written to work in any version of *Word*. For reference, more than twenty macro viruses replicate only under German *Word*, and over fifteen require the Chinese or Taiwanese versions to work.

Iceland, with its distant location and small population, has fared well in the virus field: widespread infections are rare, as are new viruses. For example, there was a large-scale outbreak of the J&M virus in Iceland during 1994, but no new viruses had been reported there in the two years previous. J&M was probably brought into Iceland in a portable PC infected while its owner was travelling in Eastern Europe.

A large number of the infections found in Nordic countries arrive via the Internet. Here, the Internet has been popular for years; indeed, Finland and Iceland have more net-users per capita than any other country in the world. This also means that net-based infections arrive very quickly in the North.

The most common Internet-connected infection type we see is a *Word* document sent via email attachment. The second most common is downloading x-rated games, animations or crack programs from the Usenet alt.binaries newsgroups.

Another common infection route originates in Russia and the former Eastern block countries. There is a lot of trade between these countries and Sweden and Finland, increasing the possibility of receiving a virus from these countries. Remarkable numbers of new viruses are written there, particularly in Russia. Norway has different contacts, with its large oil industry; and Denmark, being part of the European continent, gets frequent infections from the middle-European countries.

In the last few years, the situation in Nordic countries has become both better and worse: though we no longer see many large-scale infections, the total number of incidents has risen, due to macro viruses. According to statistics, however, the use of anti-virus software is commonplace: up to 80% of companies have a licence for some type of anti-virus program. This is a positive indication, giving us hope for the future.

TUTORIAL

Word Viruses: A Generic Guide

Over recent months, the number of new *Word* macro viruses being received by anti-virus companies has been increasing dramatically. Mirroring this, unusually for computer viruses, is the number of virus reports from the real world. Joe Wells' latest WildList (December 1996) gives mention to 22 *Microsoft Word* macro viruses – it also includes Laroux, the first *Excel* virus (see *VB*, August 1996, p.9), but this article will concern itself only with macro viruses which work in *Word 6.0* and *7.0*, as these are currently the major problem.

A further level of complexity is introduced when one considers that by the time you read this, *Office 97* should actually be on the market. *Office 97* includes (unsurprisingly) new versions of all the *Office* products, which in turn include *VBA* (*Visual Basic for Applications*) version five, a macro language very different from the current *WordBasic*.

VB readers may have noticed a certain level of similarity in recent analyses of *Word* macro viruses. Indeed, it is true to say that often, if instructions for removal are given, they will appear very similar from one such virus to the next. This article will describe attributes, often similar, of these viruses, and an improved technique for their manual removal.

Auto Macros

Many known macro viruses utilise so-called 'Auto' macros to allow themselves to receive control. *Word* is generous, and offers several such macros to the programmer:

- AutoOpen: executed when *Word* opens a document
- AutoExec: executed when *Word* is started
- AutoClose: executed when *Word* closes a document
- AutoNew: executed when *Word* creates a new document
- AutoExit: executed when *Word* exits

The existence of these macro 'hooks' is far from the only feature in *Word* that makes macro viruses possible. More and more virus authors are realising that Auto macros are not necessary to produce a perfectly functional macro virus. However, they are often seen – this is because Concept used AutoOpen, so the technique passed into virus writer lore. Far more powerful is the feature described in the next section.

Function Interception

It is a powerful, if somewhat foolish, feature of *Microsoft Word* that user-level macros are allowed to intercept what ought to be system-level features.

For example, if a user loads a document which contains a macro called 'InsertBreak', that macro will be installed as the handler for the option 'Break' on the 'Insert' menu (this will often be referred to as 'Insert/Break', representing the path taken down through the menus).

The new macro will also be called if the user accesses the menu option via a keyboard shortcut. The user is not warned that this dangerous substitution has taken place, and this fact plays a major role in the spread of macro viruses.

It is an interesting aside to point out that when a new macro is created with the same name as one of the standard built-ins, *Word* places lines of code into the new macro to call the standard handler. For example, if a macro called 'InsertBreak' is created, it will be initialised with the following code:

```
Sub MAIN
Dim dlg As InsertBreak
GetCurValues dlg
Dialog dlg
InsertBreak dlg
End Sub
```

This makes the work of the virus author who wishes to hook such macros seamlessly that much easier. He can then write his code around this, and suddenly he'll find that his mother's brother was christened Robert [*in the unlikely event that he didn't already know this... Ed.*].

Many currently extant *Word* viruses perform part of their work using macros called 'FileSave' or 'FileSaveAs'. Of course, these macros are executed when the user selects File/Save or File/SaveAs respectively. The standard viral behaviour at this point is to copy the macros that make up the virus into the document which is about to be saved, modify the file type to template (more on this point later), before executing the standard behaviour of the relevant menu option.

Initially, this can appear somewhat similar to the hooking of interrupts by DOS viruses; however, any analogies here would be misplaced. This interception of *Word* functions is much simpler, and in some ways more powerful.

Global Template

The phrase 'Global Template' crops up frequently when macro viruses are analysed. The term refers to the location where *Word* stores a given installation's configuration: the toolbars, customisations, styles, and macros which a user expects to see when he starts *Word*. For example, *Word* can be set up to italicize text when F7 is hit. On most copies of *Word*, the chances are that this action will activate the spelling checker – the customisation mentioned above would be stored in the Global Template.

In most versions of *Word*, the Global Template is called NORMAL.DOT (although this is not the case in some Far-Eastern versions). This file can be found in various locations (dependent on the version of *Word*), but in English-language versions, it is to be found in a directory called 'TEMPLATE' (*Word 6*) or 'TEMPLATES' (*Word 7*).

With *Word* versions 6.0 and 7.0, the only way for a macro virus to 'install' itself onto a PC is to copy its macros into the Global Template. Once this is done, the virus has the opportunity to infect all documents used. In some ways, this can be considered analogous to a Boot Sector virus copying itself onto the hard disk: after this is done, it can (if it so chooses) infect any diskette used.

Words and Documents and Templates, Oh My!

Readers will notice that throughout this article the author has been referring to 'documents'. The truth of the matter is that you will never see a Document that contains a macro virus. There is a clear distinction, as we will see, between Documents with a capital D and documents without it…

Some background: *Word* has two distinct primary file types, Documents and Templates. The first of these is the standard one for saved documents, and usually has the file extension DOC. The latter can contain anything that the former can, as well as macros, AutoText entries, toolbar buttons, and customized menus and shortcut keys. These are conventionally saved with the file extension DOT. However, *Word* does not enforce this extension, and Templates can be saved with any extension the user (or indeed the virus) wishes.

It can therefore be seen that when a macro virus infects a document, it must, after it has copied over the macros, arrange for that document to be saved in the form of a Template. This is a trivial matter, as is keeping the filename extension as DOC. However, some viruses come unstuck in that they allow *Word* to keep its default location for such templates, which is within the 'TEMPLATE' or 'TEMPLATES' directory mentioned above. As soon as the document type is changed to Template, *Word* is keen to save it in that directory.

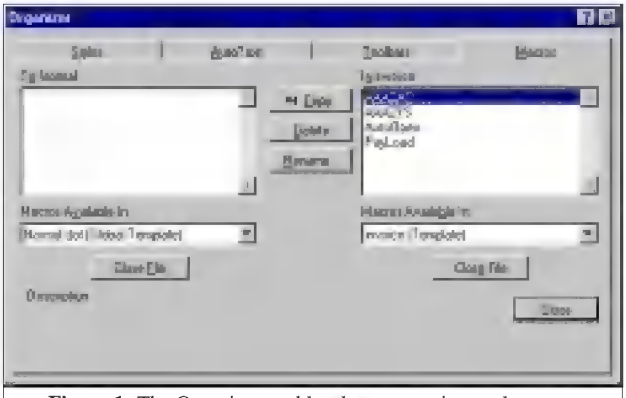

Figure 1: The Organizer enables the user to view and remove macros from documents. Here, the document in the right-hand window (invoice) can be seen to be infected with Concept.

So, if you notice documents are being saved in this directory, as opposed to where you want to put them, it is most likely because their type is being carelessly changed to Template. In turn, one of the most likely causes of this is a macro virus.

Removing a Macro Virus

Before going on to describe this in great detail, one point is worth noting – the fact that attempting to disinfect large numbers of infected documents manually will take a very long time, and leave the user open to a considerable risk of making mistakes.

Many modern anti-virus products can disinfect macro viruses with a high degree of success: the user should always try the automatic route before attempting manual cleaning.

Almost all the macro viruses in existence at the time of writing leave Tools/Macro unhooked – that is to say, when that menu option is chosen, it is not subverted by the virus, and the user is able to view the standard dialog unhindered.

However, be aware that if you load a document containing a macro called 'ToolsMacro', even if you do so with the Auto macros turned off or whilst holding down left shift, the menu option Tools/Macro will be subverted.

Therefore, a preferable technique is to use the Organizer. Start with a clean *Word* installation (simply delete the Global Template file before starting *Word*). Do not load any files; go immediately to File/Templates. Ignore the first dialog; press the 'Organizer' button. This brings up another dialog, with four tab selectors, the last of which is 'Macros'. Select this, and the display should look similar to that in Figure 1.

Press one of the 'Close File' buttons, and it becomes 'Open File'. Press this, and choose the file you wish to examine from the browser. If, then, the single-line combo box above the button reads 'Normal.dot (Global Template)', the file is not a Template. If it is, the box will read '<filename> (Template)', and the larger list box will display a list of macros held within the template. Macro names can be selected and deleted at will, and closing the file will commit the changes.

If the virus has used customisations to hook a macro to a keypress, the customisation will be left in place – errors will be generated when the document is used if the customisations are not also removed by hand.

This technique leaves less room for error than the traditional methods. At no point are macros within the document under investigation given the opportunity to take control.

Conclusion

Any paper giving 'general' information about viruses would be better accompanied by an expiry date: unfortunately, at the time of writing, the author is unlikely to know when that will be. Undoubtedly, new viruses will appear which will cause additions, but at this stage it is difficult to see how the generic disinfection instructions will be forced to change.

PRODUCT REVIEW 1

AVAST!

Dr Keith M Jackson

AVAST! is a 'set of anti-virus programs' which provides a raft of anti-virus features. The product can also be used for network protection; however, these abilities have not been tested here. *AVAST!* was provided for review on two 1.44MB floppy disks. Unfortunately, the first had a permanent data error, but the developers quickly provided another to rectify the problem. Also included in the package were four floppy disk labels pre-printed with the *AVAST!* logo: useful for clearly marking backup copies of the software.

Component Parts

I have reviewed *AVAST!* for *VB* before (February 1995): its basic concept of providing several unique anti-virus programs in one package has not changed much since then. However, the *Windows* programs have been greatly developed.

The package provided for review included seven (yes, seven!) DOS, and three *Windows*, anti-virus programs. The user is spoilt for choice, and also probably confused because the developers use strange terms for the functions provided by their programs. For instance, scanning is called 'locating'. Thus, the DOS scanner is called LGUARD (LocateGUARD). Other scanners are also included with *AVAST!*, along with a *Windows* version of LGUARD (LGW).

The other programs included with the product are: DOS and *Windows* versions of the *AVAST!* checksum verification program (AGUARD and AGW), coupled with memory-resident programs which scan files before execution (RGUARD), monitor program activity (FGUARD), and interface the memory-resident programs to *Windows* (FGW).

Finally, *AVAST!* includes three other programs: BGUARD, to repair boot sectors (why is this functionality not just placed inside the scanner?); VGUARD, which only scans for 'common viruses'; and SGUARD, which verifies checksums for a few 'important' locations.

Documentation

The documentation provided, an A5, 204-page manual, explained most of the available programs in reasonable detail. The manual was thoroughly indexed, and documented the errors produced by *Windows*-specific programs, but failed to provide this level of detail for DOS programs.

The last time I reviewed *AVAST!* for *VB*, I stated that the section of the manual entitled 'The Problem of Computer Viruses' was 'particularly well written'. It still is. The manual contains a description of a few viruses, though the total, 28, looks feeble these days.

Installation

Installing *AVAST!* is a two-part process: the DOS components come first, and the *Windows* components are then added. *AVAST!* for DOS was easy to install: the installation program asked for the name of the subdirectory in which to place its files. These are then copied across to hard disk. AUTOEXEC.BAT is (optionally) modified to include the correct calls to the memory-resident programs, and the install program offers to create a rescue diskette.

The *Windows* install program is more complicated. It first offers to scan all local disks; then, after the *Windows*-specific files have been installed, to 'save information on all files on disks' – the program was offering to create a database file of checksums. That took a while, but eventually I was asked if the database file (AGUARD.DAT) should be written to disk; then AUTOEXEC.BAT is again modified.

When *Windows* installation was almost complete, it reported that nineteen files had been copied, and instructed the user to run SETUP.EXE from *Windows*. But SETUP.EXE does not exist. Further, NEWS.DOC states that the install program is for DOS and *Windows*. So why did I just run both?

I tried again with the content of both *AVAST!* floppy disks copied into a single subdirectory: this time the installation program reported that 49 files had been copied, and did not instruct the user to run SETUP. Something is wrong with the install process. At a guess it's because changes have been made which have not been applied consistently throughout.

When this was finished, I found that *AVAST!* had installed three memory-resident programs (LGUARD, RGUARD and FGUARD) in AUTOEXEC.BAT, and created five *Windows* icons (AGW, FGW, LGW, SGW and a de-install program).

Checksumming

The first time the *AVAST! Windows* checksummer (AGW) was run, it created a checksum file for the files on my test PC's hard disk. This took 1 minute 17 seconds. SYS, OV?, BAT, EXE, COM and VPS files have checksums created by default.

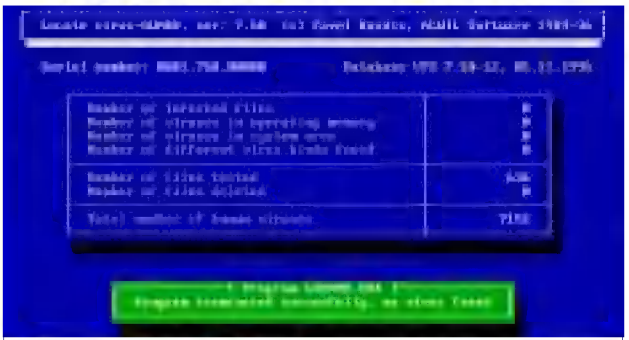

Figure 1: LGUARD is *AVAST's* main virus detection component.

AGW always took 1 minute 12 seconds to verify checksums for my test PC's hard disk, though it proved hard to tell when AGW had finished verification. Oddly, it fills a window with a line-by-line list of files it is verifying – when this is full, it adds a scroll bar, leaving the original text alone. Extra lines are added invisibly at the bottom of the full window. Scrolling the window makes it possible to view all the text.

I found AGW's interface confusing. It is necessary to figure out why 'Add' and 'INI' operations are available for file selection, and use them to select the files you want saved to disk. When AGW is restarted, it says 'Changes in the content to the system area. System area on disk C: has changed. Is such a change OK?' Reply 'no': it checks the disk. Reply 'yes': it still checks the disk. When it is next executed, AGW produces the same message, no matter what the reply.

I even erased all *AVAST!* database files created on disk, and started from scratch. AGW *still* said 'system area on C: has changed'. This confused me: even after 20 minutes trying to figure it out I was no wiser. Nor did AGUARD for DOS make matters clearer: this program said 'boot sector at disk C: was changed', seemingly no matter what I did.

Scanner Operation

The *Windows* scanner (LGW) commences scanning as soon as it is executed. This takes ages. The scan was difficult to terminate: it used so much of *Windows'* resources that the mouse pointer only moved across the screen intermittently. When scanning memory the mouse is unusable, though when scanning files the movement improved somewhat.

I cannot see any reason for commencing a scan immediately the program is executed. Either let the user press 'Go', or rely on a scheduler. Having LGW recommence the last defined scan whenever it is reactivated is a bit disconcerting.

AVAST! is the first anti-virus product to talk to me! [*The stress of many years writing for VB began to show in early 1997... Ed.*] When the *Windows* scanner (LGW) detected a virus the sound card activated and a voice said 'Attention please, your computer is infected by a virus'. Happily, this only happens for the first virus found. Subsequent infected files just produce an onscreen message; no unearthly voices.

Scanning Speed

In its default state, running under DOS, LGUARD scanned my test PC's hard disk in 1 minute 24 seconds (441 of 1584 files scanned). With memory scan removed, this reduced to 1 minute 18 seconds. If all parts of each file were scanned, scan time increased to 3 minutes 45 seconds. In comparison, *Dr Solomon's AVTK* did the same scan in 1 minute 2 seconds; *Sophos' SWEEP*, in 2 minutes and 1 second.

For all the above tests, files with extensions COM, EXE, SYS, OV?, BIN, DLL, DO? and BAT were inspected. The version of *AVAST!* provided for review stated it could currently scan for 7152 viruses.

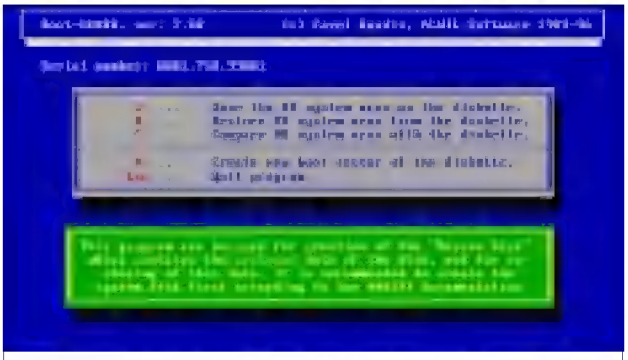

Figure 2: BGUARD allows the user to remove boot sector viruses from diskettes and from the hard drive.

When LGW was used to scan a hard disk, the scan times went into another dimension. A scan which was allowed to complete with the PC used for nothing else took 5 minutes 9 seconds. If any other *Windows* component was activated, this increased pro rata. Scanning all files increased scan time to 10 minutes 56 seconds – on the verge of unacceptable.

I would be happier with LGW if it really was a background scanner which did not affect the PC, scanning invisibly in the background. However, remember my comments above that LGW used up so much of *Windows'* resources that moving the mouse pointer cleanly was nigh on impossible, and therefore even closing down the scan was quite difficult.

Virus Detection

Against the viruses in the In Wild test-set, using default settings, LGW detected all 286 samples. Against the viruses in the Standard test-set, again using default settings, it detected 262 of the 265 samples (99%). The sole sample of EvenBeeper and both samples of Cruncher were not detected. Even activating the option to scan all parts of all files did nothing to improve detection.

When tested against the polymorphic virus samples, *AVAST!* detected all 5500 test samples. In similar vein, all twenty boot sector test samples, and all 29 *Windows*-specific test samples were correctly detected.

LGUARD exhibited the same ability. In fact, the main difference between the two scanners seemed to be that the *Windows* version was *much* slower (see above), and ran out of space in its report file when many viruses were found (Notepad was used to view the log – it was simply too big).

False Positives

Against the *VB* false positive test-set (5500 executable files occupying around 600MB), LGW found none of the files to be falsely infected. LGW is cautious when faced with scanning an entire CD-ROM (on which the *VB* test-sets are stored) – it asks 'Are you really sure that you want to test the files on the CD-ROM disk?', and awaits confirmation before proceeding on this long scan. A well-thought-out minor feature.

Testing the entire CD-ROM for false positives threw up lots of 'Error while working with File' messages, several 'Access Denied' messages, and a few 'Open Error' messages. I never figured out why these were displayed, but it did not detract from the lack of false positives exhibited by *AVAST!*.

VGUARD

Last time, I was baffled by VGUARD, a 'minimal' scanner, which looks for a very short list of 21 viruses. I still am. *AVAST!* contains a scanner that detects everything thrown at it: given this excellent result, what's the point of including a program that is nowhere near as good at detection? *ALWIL* has since stated that the program would probably soon be removed; and has not been updated for some time.

VGUARD detected 20 of the 286 samples in the 'In the Wild' test-set, and only eight of the 265 Standard samples (99%). It missed all the polymorphic, macro and boot sector samples.

Curiously, VGUARD claimed to 'immunize' 28 of the In the Wild viruses, and 43 of the Standard set. How something not detected as infected can be immunized is beyond me. Likewise, many polymorphic samples are flagged as 'immunized against virus 648' – the developers state that this means the time-stamp is set to 62 seconds. However, none of the 5500 test samples were detected as infected by VGUARD.

Memory-resident Software

AVAST! states that its memory-resident scanner uses the same database as the main scanner(s). However, *AVAST!* only scans for viruses when programs are executed rather than when they are read from disk, so virus-infected files can be copied, but not executed. It is thus beyond the scope of this review to test the detection rate of this component.

The memory-resident software provided with *AVAST!* does exhibit problems. When FGUARD detects a change to a proscribed type of file, a red box pops up onscreen, giving various options. No matter which option was selected, the software locked up, and a reboot was necessary. Even if this is cured, FGUARD pops up too often for it to be usable.

RGUARD and FGUARD required only 8.3KB (the manual claims 6.2KB) and 10.1KB respectively. The *Windows* version (FGW) refused to perform. Whenever its icon was activated, a message box appeared stating 'Programs FGUARD or RGUARD for Windows do not reside in memory, or either of them is not mutually compatible with the FGW'. The DOS programs RGUARD and FGUARD were present, so I can only conclude that the compatibility hint was correct.

I measured the overhead introduced by the memory-resident programs by timing how long it took to copy 40 files (1.25MB) from one subdirectory location to another. This took 13.1 seconds without memory-resident software, and 10.7 seconds with the software installed. I can think of no mechanism whereby the introduction of *extra* memory-resident software can speed things up. It is a seemingly impossible (but nonetheless repeatable) result.

The Rest

AVAST! utilizes a program called BGUARD to disinfect boot sector viruses from diskettes and manage the workstation recovery disk. I did not review these. SGUARD just verifies checksums for 'important' locations, and the same criticisms apply to this program as I outlined above for VGUARD. Why not just use AGUARD? This is a very good checksummer.

Conclusions

AVAST!, in recent reviews, detected 100% of all viruses: this month the detection rate is only 99%. Why does it now miss three viruses from the 'Standard' test-set? It's most odd.

There are some problems, albeit only minor ones, associated with installation. I have previously criticised *AVAST!* for its complexity as it was provided as a set of individual programs. This seemed confusing. *Windows* has made this worse: the presence of GUI versions increases the number of executables.

Last time I reviewed *AVAST!*, I said 'Given revision of the user interface structure, and extra documentation, the developers of *AVAST!* could well have a world-wide winner on their hands'. I think that prediction has come true, also in view of the fact that other developers want to badge this scanner. Indeed, my overall conclusion about *AVAST!* remains the same as last time: 'It's one of the best performing anti-virus programs that I've reviewed in a long while.'

AVAST! is a traditional anti-virus software package with all the components you could ever need (and a few you don't), coupled with some of the best virus detection you are likely to find anywhere.

Technical Details

Product: *AVAST! v7.5*, serial number 0001.750.99001.

Developer/Vendor: *ALWIL Software*, Lipi 1244, Prague 9 19300, Czech Republic. Tel +42 2 685 5961, fax +42 2 685 5624, email: baudis@alwil.anet.cz.

Distributor UK: *Lance Premier Services*, Britannia House, 4-24 Britannia Street, London WC1X 9JD, England. Tel +44 171 681 8610, fax +44 171 681 8615.

Availability: General requirements are an *IBM* PC, XT or AT with at least 256KB RAM and DOS v3.3 or higher – I believe these are for the DOS-specific components. *Windows* components (SGW, LGW and AGW) require an 80386SX, and at least *Windows 3.xx*. LGW needs at least *Windows 3.1* (in enhanced mode), and DOS version 3.3 or above.

Price: One-year licence includes twelve updates, and number of workstations equals number of ordered licences. 1–10, £59.90; 11–25, £18.00; 26–50, £14.00; 51–100, £11.00; 101–250, £9.00; 251–500, £6.50; 501–1000, £5.00; 1000+, £4.50.

Hardware: 33MHz 486 PC with 12MB RAM, one 3.5-inch (1.44MB) and one 5.25-inch (1.2MB) floppy disk drive, 1GB hard disk space, running under *MS-DOS v5.0* and *Windows v3.1*.

Viruses used for testing: Where more than one variant is used, the number of samples is shown in brackets after the virus name (if greater than one). For a complete explanation of each, and the nomenclature used, refer to the list of viruses published in *VB*. A listing of the boot sector viruses can be found in *VB*, March 1996, p.23. Listings for the other test-sets are in *VB*, January 1996, p.20.

PRODUCT REVIEW 2

ThunderBYTE for NetWare

Martyn Perry

This month we take a detailed look at *ThunderBYTE for NetWare* (*TBAVNW*) from *ESaSS BV*. At the time of writing, it was in the final stages of beta testing, but the authors felt confident enough to let us put the product through its paces.

Presentation and Installation

The product arrived on a single 3.5-inch floppy disk, and without documentation. *TBAVNW* is licensed on a per-server basis with no other provisions. There is no separate workstation software provided: any workstations would use another product, presumably here *ThunderBYTE for DOS*, which is of course available separately.

The installation requires the following minimum versions of *Novell's* CLIB.NLM:

- *NetWare* v3.11/3.12: v3.12h
- *NetWare* v4.0x: v4.01c
- *NetWare* v4.10: v4.10

With the correct version in place, installation can proceed. Once a network drive is mapped to SYS, the program INSTALL.EXE can be run from the distribution diskette.

This creates the main directory, SYS:\TBYTENW, the quarantine directory SYS:\TBYTENW\VIRUS, and the log directory SYS:\TBYTENW\LOG. The main directory contains the NLM and the virus definition files. In addition, TBAV.NCF is copied into SYS:\SYSTEM to provide an easy way to start the NLM. This can be called from a modified AUTOEXEC.NCF to allow the NLM to be loaded as the server starts.

Getting Started

The NLM offers three command-line options;

- /SD: disables the 5 millisecond delay between each file being scanned
- /I: disables exact virus identification
- /MTD: specifies the maximum subdirectory depth

The first option increases the scan speed considerably, at the expense of similarly increasing CPU utilisation. The second increases the scan speed on infected systems by reporting a virus as related to a particular family, rather than bothering to identify the precise variant. The third option is /MTDxxx, where xxx is a number from 10 to 100. This allows the user to change the maximum number of levels in the server's directory tree – the default is 25. If the NLM finds directories lower down the tree than has been set, these will be skipped and a warning message displayed when the scan is complete.

Administration

Scanner administration is performed from the server console. The program adopts the same menu format as most other *Novell* utilities. The main menu gives options to initiate a scan of the server, configure the scanner, display the Monitor screen, and show the Virus Library.

When the scanner is running, information is displayed in three areas: the top section gives the version of the virus definition file (date and number of viruses detected). There is also a total count of directories, all files, and scanned and infected files. The middle section shows the names of the current volume, directory and file being scanned, and the bottom section displays any infected files by name along with the action being used to deal with it.

The top of the Monitor screen gives version information on the virus definitions. The rest of this section gives the utilisation of the server and the number of connections in use. The middle section shows the status of the options selected for the scanner. The bottom section covers details of the last infected file, what caused the file to be scanned and the subsequent action taken to deal with the infection.

The Virus Library allows the user to browse through the list of viruses the scanner can detect. For each sample, the display shows known variants, if and where the virus goes memory-resident, if it has a destructive payload, and the types of files infected. Boot sector virus details are included for information only: the scanner does not scan for boot sector viruses.

A screen-blanking option can also be enabled: this blanks the console screen three minutes after it was last used. The screen can be redisplayed by pressing any key.

TBAVNW has two modes of scanner operation: immediate and on-access (real-time). An immediate scan allows the administrator to start and stop the scan from the server console. The screen displays the progress of the scan, including the files checked and any viruses found.

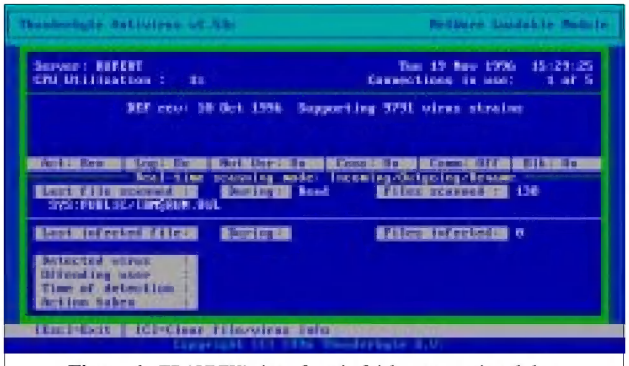

Figure 1: *TBAVNW's* interface is fairly conventional, but displays all the necessary information, and is easy to use.

On-access scan settings allow the administrator to choose whether or not to check incoming files, outgoing files, or both or neither. In addition, files can be scanned when they are renamed. Further, it is also possible to disable on-access scanning completely.

No facility is provided for scheduled scanning. The rationale behind this, according to *ThunderBYTE's* developers, *ESaSS BV*, is that real-time scanning is sufficient to prevent viruses reaching the server, which should render any scheduled scanning features superfluous.

Configuration Options

For both immediate and real-time scans, various selections can be made by selecting 'Configure ThunderBYTE' from the main menu. These include:

- File extensions: 386, APP, BAT, BIN, COM, DLL, DOC, DOT, EXE, OVL, OVR, SYS, VBX, XLC, XLS, XLM, and XTP. There is no provision to modify this extensions list.
- An option to move infected files to the quarantine directory. If this is chosen, the immediate scanner will scan this directory and add the files found to its detection list. Other than that, there is no provision to exclude areas such as directories or files from the scan. The developers take the view that this could make the server vulnerable.

A separate menu option allows the selection of actions to be taken on detection of a virus. They are:

- move infected files off-line, i.e. to the quarantine directory, with a further option to use incrementing numeric names for the moved files
- purge infected files using *Novell's* Purge utility
- rename infected files

Alert Management

TBAVNW offers various methods of alerting administrators to the presence of a virus, namely:

- simply displaying a message on the server console
- notifying the offending user via *Novell's* Send/Broadcast facility
- notifying a predefined group of users, also via *Novell* messaging
- selecting a print queue to which to send an alert message

A further facility is available for multi-server environments. This will allow a server to send a message to another server acting as a Communications Hub. Only one server may be designated to be operating as the Communications Hub at any one time.

If any member of the group to be alerted is not logged in when an alert is issued, the information is stored in the file TBGRPSAV.DAT until such time as the user does log in. This allows the data to be retrieved if the NLM is unloaded

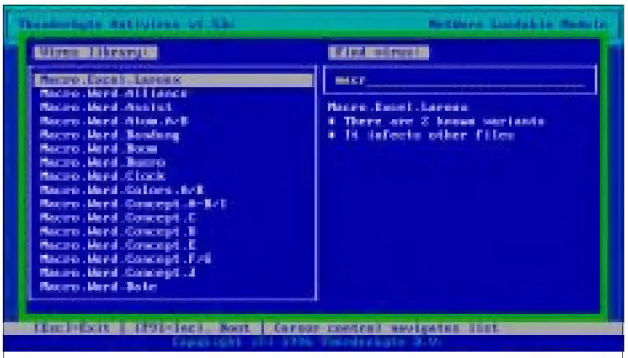

Figure 2: *TBAVNW's* virus encyclopædia contains useful, albeit very limited, information on the viruses it detects.

or the server is downed. The file TBGRPSAV.DAT can also, optionally, be used to store information about the last virus detected, and about the total number of infected files which were found.

Reports, Activity Logs, and Updates

TBAVNW uses four log files. These are stored in the directory SYS:TBYTENW\LOG, and are created automatically, as and when they are needed.

The four log files are: TBERROR.LOG (which stores error messages), TBREALTI.LOG (which logs virus incidents generated by the real-time scanner), TBSCAN.LOG (which keeps the results of manual scans), and TBVIRUS.LOG (which stores the name of each infected file that has been moved to SYS:\TBYTENW\VIRUS, the quarantine directory). This last file logs the file's original location as well as the name of the virus found within.

Updates involve simply replacing the signature file (called NVC.DEF) with an updated version. The scanner must then be restarted for the new signatures to take effect.

Detection Rates

The scanner was checked using the usual three test-sets: In the Wild, Standard and Polymorphic (see summary for detail). The virus signature list used was dated 10 October 1996, and contained knowledge of 9791 virus strains. The viruses that were not detected were identified by using the 'move infected files' option and listing the files which were left behind in the virus directories.

The tests were conducted using the default scanner file extensions supplied. The Standard test result was an impressive 100%. The In the Wild set produced 98%, failing only on samples of Wazzu, Satan_Bug, and Imposter. The Polymorphic test yielded a very creditable 98.4%: misses here were mainly samples of Satan_Bug.

Real-time Scanning Overhead

To determine the impact of the scanner on the server when it was running, 63 files of 4,641,722 bytes (EXE files from SYS:PUBLIC) were copied from one server directory to

another using *Novell's* NCOPY. NCOPY keeps data transfer within the server itself and minimises network effects. As usual, the directories used for the source and target were excluded from the virus scan to avoid the risk of a file being scanned while waiting to be copied.

To compensate for the different processes which occur within the server, the time tests were run ten times for each setting and an average was taken. The tests (see summary for detailed results) were:

- NLM not loaded – this establishes the baseline time for copying the files on the server
- NLM unloaded – this is run after the other tests to check how well the server is returned to its original state
- NLM loaded, no files entering or leaving the server, and no scan. This tests the impact of the scanner loaded in its quiescent state with neither a real-time nor an immediate scan in progress.
- NLM loaded, files entering but not leaving the server, and no scan. This test records the impact of running the real-time scan on incoming files without the immediate scan.
- NLM loaded, files entering but not leaving the server, and no scan. This shows the real-time scan effect on incoming and outgoing files.
- NLM loaded, files entering and leaving the server, and immediate scan. This shows the incremental effect of running the immediate as well as the real-time scan.
- NLM loaded, files entering and leaving the server, and immediate scan with file delay suspended. This shows the effect if the 5 millisecond delay was not in operation.

The impact of the scanner begins to take effect when the real-time scan is selected. This is further affected when the immediate scanner is run.

The default selection has a 5 millisecond delay between file accesses. If this is removed, the scan speed increases dramatically, but at the expense of other file activities. Day-to-day use of the scanner requires this delay to maintain server performance.

The residual overhead when TBAVNW is unloaded is due to the CLIB and Streams NLMs remaining loaded.

Conclusion

The product is straightforward to install, and the detection rates are of the high level that we have come to expect from *ThunderBYTE*. The version tested had no F1 help support, but the configuration options were generally easy to select.

The Communications Hub provides a measure of inter-server communication, but no administration facilities for multi-server management. The decision not to include a scheduler is an interesting one, since most *NetWare* scanners provide some level of timed scan.

My one concern is the fixed choice of file types. I would prefer to have the file extensions list user configurable with the default selection stored in the software. This would allow new file types (e.g. *Lotus* files) to be added by the user without waiting for a program update.

ThunderBYTE for NetWare

Detection Results

Test-set[1]	Viruses Detected	Score
In the Wild	335/342	98.0%
Standard	511/511	100.0%
Polymorphic	9841/10000	98.4%

Overhead of On-access Scanning:

The tests show the time taken to copy 63 EXE files (4.6MB). Each test is performed ten times, and an average is taken.

	Time	Overhead
NLM not loaded	4.0	–
NLM unloaded	4.1	4.0%
NLM Loaded		
No files entering/leaving server, no scan	4.0	0.0%
Files entering but not leaving server, no scan	7.0	75.0%
Files entering/leaving server, no scan	8.6	115.0%
Files entering/leaving server, scan	14.5	262.5%
Files entering/leaving server, scan, /SD command	28.6	615.0%

Technical Details

Product: *ThunderBYTE for NetWare v1.53.*

Developer/Vendor: *ESaSS BV*, Saltshof 10-04, NL-6604 EA Wijchen, The Netherlands. Tel +31 24 64 88 555, fax +31 24 64 50 899.

Distributor UK: *CPL ThunderBYTE*, Berkhamstead House, 121 High Street, Berkhamstead, Hertfordshire HP4 2DJ, England. Tel +44 1442 870161, fax +44 1442 870148.

Price: The pricing structure of this new product was unavailable at the time this issue went to print; for information, contact the vendors either in the Netherlands or the UK.

Hardware Used:
Server: *Compaq Prolinea 590* with 16MB RAM and 2GB of hard disk, running *NetWare 3.12*.
Workstation: *Compaq 386/20e* with 4MB RAM and 207MB of hard disk, running DOS 6.22 and *Windows 3.1*.

[1]Test-sets:
In the Wild File, In the Wild Boot Sector, and Polymorphic – see *Virus Bulletin*, October 1996, p.17. Standard – see *Virus Bulletin*, November 1996, p.23.
For a complete explanation of each virus, and of the nomenclature used, refer to the list of viruses published regularly in *Virus Bulletin*.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues) including first-class/airmail delivery:

UK £195, Europe £225, International £245 (US$395)

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire, OX14 3YP, England

Tel 01235 555139, International Tel +44 1235 555139
Fax 01235 531889, International Fax +44 1235 531889
Email: editorial@virusbtn.com
World Wide Web: http://www.virusbtn.com/

US subscriptions only:

June Jordan, *Virus Bulletin*, 590 Danbury Road, Ridgefield, CT 06877, USA

Tel +1 203 431 8720, fax +1 203 431 8165

END NOTES AND NEWS

***Softbank Corporation* is to invest US$31 million for a 35% stake in anti-virus software multi-national *Trend Micro Devices*.** *Softbank* plans to encourage hardware makers world-wide to use *Trend's* anti-virus software in their systems, and to integrate its software in memory boards. Information on the incentive can be found at *Trend's* Web site; http://www.trend.com/.

***Sophos Plc's* next round of anti-virus workshops** will be on 19/20 March 1997 at the training suite in Abingdon, UK. The company's training team is also hosting a Practical *NetWare* Security course on 13 March 1997 (cost £325 + VAT). Information is available from Julia Edwards, Tel +44 1235 544028, fax +44 1235 559935, or access the company's World Wide Web page; http://www.sophos.com/.

InfoSecurity 1997 will take place at the Olympia 2 (London, England) from 29 April–1 May 1997. The event is planned to address all aspects of IT security in the business environment, and many anti-virus developers will be present. For information, contact Yvonne Eskenzi on Tel +44 181 449 8292, or on the Web at http://www.infosec.co.uk/.

Reflex Magnetics is presenting **computer security courses**: a Live Virus Experience will be held on 19/20 February 1997, and The Hacking Threat from 4–6 March 1997. The venue for both is *Reflex's* premises in London, England. For further information, contact Phillip Benge at *Reflex Magnetics*; Tel +44 171 372 6666.

The *Computer Security Institute* (*CSI*), at its recent Computer Security Conference, has presented a **Lifetime Achievement Award to Charles Cresson Wood**. Also at the conference, the award for Information Security Program of the year went to *Detroit Edison*. The *CSI's* next conference, NetSec '97, will be held from 9–11 June 1997 in San Francisco. For more information, contact the *CSI* on Tel +1 415 905 2626, or visit the Web site; http://www.gocsi.com/.

McAfee is the latest company to extend its encryption software, with *PCCrypto*, a product which, the company claims, allows users to **secure both their desktop data and their Internet/intranet mail communications.** Contact Caroline Kuipers for information; Tel +44 1344 304730, email caroline_kuipers@cc.mcafee.com.

***SecureNet 97* will take place on 20/21 March 1997 in Cannes, France**; speakers include such well-known anti-virus 'names' as Fred Cohen, Vesselin Bontchev, Klaus Brunnstein, and Eugene Spafford. Information is available from Alex Verhoeven at *Elsevier Advanced Technology*; Tel +44 1865 843654, fax +44 1865 843971, email a.verhoeven@elsevier.co.uk.

Dr Solomon's Software Ltd (formerly *S&S International*) is presenting **Live Virus Workshops** at the *Hilton National* in Milton Keynes, Bucks, UK on 19/20 February and 25/26 March 1997. Details from Melanie Swaffield at *Dr Solomon's*; Tel +44 1296 318700, Web site http://www.drsolomon.com/.

Software developer ***Precise Publishing* has announced the release of *MacroBlaster***, a program designed to deal with the threat of infected *Word* documents. The package contains a batch scanner and cleaner, as well as real-time protection. For further information, contact the company on Tel +44 1384 560527.

The ***First Annual International Banking and Information Security Conference*** will take place from 19–21 February 1997 in New York City. Information from the *NCSA*; Tel +1 717 241 3226, or visit the Web site; http://www.ncsa.com/.

The proceedings of the sixth *VB* conference are still available; price £50 + p&p. To order, contact Conference Coordinator Alie Hothersall; Tel +44 1235 544034, email alie@virusbtn.com.